Refine Search

Search Results -

Terms	Documents
db2 near tablespace	6

Database: US Pre-Grant Publication Full-Text Database; US Patents Full-Text Database; US OCR Full-Text Database; EPO Abstracts Database; JPO Abstracts Database; Derwent World Patents Index; IBM Technical Disclosure Bulletins

Search:

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Saturday, September 11, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L21	db2 near tablespace	6	L21
L20	L19 and (backup or back-up) near (copy or copies or replica)	29	L20
L19	L18 and (database or data with base or data adj base) near (recover or recovery or recover$)	96	L19
L18	707/204 and (tablespace or table or space or table-space)	1521	L18
DB=EPAB; PLUR=YES; OP=OR			
L17	CN-1421010-A.did.	0	L17
L16	CN-1421010-A.did.	0	L16
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR			
L15	L14 and (backup or back adj up) near (copy or copies or replicat or replicate)	5	L15
L14	L13 and (table with space or table adj space or table-space or tablespace)	39	L14
L13	L12 and table near (recover or recovery or recover$)	145	L13

L12	(database or data adj base or data with base)	484707	L12
L11	707.clas.	22340	L11
L10	707/204	1919	L10
L9	707/203	2448	L9
L8	707/202	1818	L8
L7	707/201	2361	L7
L6	707/200	3471	L6
L5	707/7	1633	L5
L4	707/102	5035	L4
L3	707/101	3479	L3
L2	707/100	5093	L2
L1	707/1	6919	L1

END OF SEARCH HISTORY

L15: Entry 2 of 5 File: USPT Feb 16, 1999

DOCUMENT-IDENTIFIER: US 5873091 A
TITLE: System for data structure loading with concurrent statistical analysis

Brief Summary Text (3):
The present invention relates to the use of database tables in computers. More particularly, the invention concerns an apparatus, article of manufacture, and process for loading, reorganizing, or recovering a data structure and simultaneously collecting various statistics concerning the data structure. Such data structures may comprise, for example, tables and/or indices.

Brief Summary Text (5):
Database applications constitute one of the most popular applications of computers today. Engineers have designed many different systems and programs to store, query, and modify information stored in database tables. Intricate manipulation of table data is performed using a database language such as "SQL". Higher level operations are typically achieved by executing various high level database commands, in a database system language such as the "DB2" product of International Business Machines Corp. Some of these high level operations include "Load", "Image Copy", "Reorg", and "Recover" operations, for example.

Brief Summary Text (6):
Typically, a database management system performs a "Load" operation to compile data from various sources, and ultimately assemble the data into a new or existing table. A "Reorg" operation copies all rows from of an existing table, stores the rows separately, empties the table's contents, and then re-loads the copied rows back into the table in a desired manner. For example, Reorg procedures are useful to eliminate unwanted space in a table, and to change the order of rows in a table to optimize the table. A "Recover" operation serves to restore a table using a backup copy. Used when a table fails for some reason, a Recovery operation reads table pages from a backup copy, and writes the pages to the failed table. A "Recover Index" operation restores an index using data from an indexed table. More particularly, the Recover Index operation replaces a failed index by scanning the indexed table, and extracting all entries from columns designated as "key" columns. The Recover Index stores these entries in the recovered index, along with the location of each entry in the underlying table.

Brief Summary Text (8):
The statistics from Runstats are useful for a number of different purposes. First, these statistics enable the database system to optimize access to the data to satisfy a query expressed in a database query language such as "SQL". For example, if the results to a query can be obtained either by scanning an entire table or by reading the table through an index, the database management system ("DBMS") might decide how to process the query based on the statistics relating to the size of the table. If the table is small, for example, it may be more efficient to scan the table rather than to traverse the index.

Brief Summary Text (9):
The Runstats operation is also useful to allow database administrators ("DBAs") to access the status of the database. For example, observing a low value for a table's clustering ratio, a DBA may conclude that the table should be reorganized.

Reorganization will resequence the rows in the table to restore their optimal order.

Detailed Description Text (4):
Database Manager

Detailed Description Text (5):
The DBMS 100 includes a database manager 102. Preferably, the DBMS 100 comprises a multiprocessing computer such as an IBM System 390 model 9672 mainframe computer. The database manager 102 may utilize an operating system such as IBM MVS, for example. And, although other database languages may be used, the DBMS 100 preferably uses the DB2 language of International Business Machines Corp.

Detailed Description Text (10):
The database manager 102 preferably includes one or more processing units, such as the illustrated load processing unit 104 and statistical analysis processing unit 106. Although not shown, a single multitasking processing unit may be used, where separate operational procedures are performed in a split-cycle fashion, for example. Nonetheless, multiple processing units, as illustrated, are preferred to maximize operation speed. Each processing unit 104/106 may comprise a microprocessor or another suitable digital data processing apparatus. In an embodiment where the DBMS 100 comprises an IBM System 390 computer, for example, each processing unit 104/106 may comprise a central processing unit ("CPU") of the computer.

Detailed Description Text (20):
This data storage medium may reside, for example, in a RAM program storage buffer (not shown) contained in the database manager 102. Alternatively, the instructions may be contained in another data storage medium, such as a magnetic data storage diskette 200 (FIG. 2). Whether contained in the database manager 102 or elsewhere, the instructions may instead be stored on another type of data storage medium such as hard drive storage (e.g., a conventional magnetic disk or RAID array), magnetic tape, electronic read-only memory (e.g. ROM), an optical storage device (e.g. WORM), paper "punch" cards, or other suitable data storage media. In an illustrative embodiment of the invention, the machine-readable instructions may comprise lines of compiled DB2 language code.

Detailed Description Text (39):
In addition to features concerning the data itself, other statistics concern the way in which the data is stored. One example is the percentage of rows that are in clustering order, called the "cluster ratio". If a table is defined with a clustering key, then rows should optimally be stored in a sequence by the value of that key. The cluster ratio is therefore the percentage of rows that are in the proper order. Another useful statistics is the percentage of space in the data structure occupied by data, i.e. "percent active" space. Of the total space allocated to a database on a storage device, the amount used by the database table is the "percent active", the rest is free space.

Detailed Description Text (40):
When the query 308 determines that the statistical analysis processing unit 106 has processed all rows of the data structure 114, task 312 stores the results of the statistical analysis in a "data dictionary" or other suitable index or catalog. As an example, the statistical analysis processing unit 106 may store the data dictionary in storage (not shown) of the database manager 102. As an alternative, the statistical analysis processing unit 106 may instead forward the data to the load processing unit 104 for storage elsewhere, such as with the data structure 114. Following task 312, the routine 352 ends in task 314.

Detailed Description Text (43):
For instance, ordinarily skilled artisans will appreciate that, in addition to

loading tables, the operational sequence 300 is similarly applicable to indices, as well as other operations such as the reorganization of existing data structures, or recovery of tables or indices.

Other Reference Publication (1):
C.J. Date, An Introduction to Database systems , 6th Ed., 1995 p. 769.770.

CLAIMS:

8. The method of claim 1, the one or more data sources comprising a backup copy of the data structure, the forming of the data structure forming the data structure by recovering the data structure from the backup copy.

17. The article of manufacture of claim 10, the one or more data sources comprising a backup copy of the data structure, the forming of the data structure forming the data structure by recovering the data structure from the backup copy.

24. The multiprocessing system of claim 19, the one or more data sources comprising a backup copy of the data structure, the forming of the data structure forming the data structure by recovering the data structure from the backup copy.

31. The multiprocessing system of claim 26, the one or more data sources comprising a backup copy of the data structure, the forming of the data structure forming the data structure by recovering the data structure from the backup copy.

Previous Doc Next Doc Go to Doc#

L15: Entry 2 of 5 File: USPT Feb 16, 1999

US-PAT-NO: 5873091
DOCUMENT-IDENTIFIER: US 5873091 A

TITLE: System for data structure loading with concurrent statistical analysis

DATE-ISSUED: February 16, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Garth; John Marland	Gilroy	CA		
John; Koshy	Fremont	CA		
Ruddy; James Alan	Gilroy	CA		
Schwartz; David Ray	Morgan Hill	CA		
Smith; Bryan Frederick	Morgan Hill	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 848006 [PALM]
DATE FILED: April 28, 1997

INT-CL: [06] G06 P 15/163

US-CL-ISSUED: 707/102; 707/7, 707/8, 707/100, 707/101, 707/707, 707/202, 707/203, 707/204
US-CL-CURRENT: 707/102; 707/100, 707/101, 707/202, 707/203, 707/204, 707/7, 707/8

FIELD-OF-SEARCH: 707/7, 707/8, 707/100, 707/101, 707/102, 707/202, 707/203, 707/204

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	5222235	June 1993	Hintz et al.	707/101
☐	5265244	November 1993	Ghosh et al.	707/1
☐	5446575	August 1995	Lysakowski, Jr.	707/104
☐	5530964	June 1996	Alpert et al.	395/709

☐	5692171	November 1997	Andres	707/1
☐	5758357	May 1998	Barry et al.	707/202

OTHER PUBLICATIONS

C.J. Date, An Introduction to Database systems , 6th Ed., 1995 p. 769.770.

ART-UNIT: 276

PRIMARY-EXAMINER: Kulik; Paul V.

ASSISTANT-EXAMINER: Homere; Jean R.

ATTY-AGENT-FIRM: Gray Cary Ware Freidenrich

ABSTRACT:

A multiprocessing system forms a data structure, such as by loading reorganizing, or recovering, while concurrently collecting various statistics about the data structure. The data structure may comprise tables and/or indices, for example. A first processing unit forms the data structure by assimilating data from one or more data sources into data rows, storing the rows in a buffer, and copying the rows from the buffer to the data structure. Concurrently with the forming step, the same or a second processing unit retrieves the rows from the buffer and applies a predetermined analysis to the rows to formulate statistics regarding the data structure.

32 Claims, 3 Drawing figures

Previous Doc Next Doc Go to Doc#

L15: Entry 4 of 5 File: USPT Apr 4, 1995

US-PAT-NO: 5403639
DOCUMENT-IDENTIFIER: US 5403639 A

TITLE: File server having snapshot application data groups

DATE-ISSUED: April 4, 1995

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Belsan; Jay S.	Nederland	CO		
Laughlin; Jeffrey S.	Nederland	CO		
Pedersen; Mogens H.	Longmont	CO		
Raicer; Robert J.	Niwot	CO		
Rudeseal; George A.	Boulder	CO		
Schafer; Charles P.	Louisville	CO		
Steele; Barbara L.	Boulder	CO		
Tomsula; Patrick J.	Arvada	CO		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
Storage Technology Corporation	Louisville	CO			02

APPL-NO: 07/ 939312 [PALM]
DATE FILED: September 2, 1992

INT-CL: [06] G06 F 15/40

US-CL-ISSUED: 395/600; 395/425, 395/800, 395/650, 364/DIG.2
US-CL-CURRENT: 707/204; 707/205, 711/113, 711/114

FIELD-OF-SEARCH: 395/600, 395/425, 395/800, 395/650, 395/400

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4627019	December 1986	Ng	395/425
☐	5124987	June 1992	Milligan et al.	371/10.1
☐	5155835	October 1992	Beisan	395/425

☐	5193184	March 1993	Belsan et al.	395/600
☐	5210866	May 1993	Milligan et al.	364/DIG.2
☐	5212789	May 1993	Ragu	395/600
☐	5239659	August 1993	Rudeseal et al.	395/800
☐	5247647	September 1993	Brow et al.	395/600
☐	5278979	January 1994	Foster et al.	395/600
☐	5287496	February 1994	Chen et al.	395/600

ART-UNIT: 237

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Pham; Cuan

ATTY-AGENT-FIRM: Duft, Graziano & Forest

ABSTRACT:

This file server system appears to the host computer to be a plurality of data storage devices which are directly addressable by the host computer using the native data management and access structures of the host computer. The file server however is an intelligent data storage subsystem that defines, manages and accesses synchronized sets of data and maintains these synchronized sets of data external from the host computer system's data management facilities in a manner that is completely transparent to the host computer. This is accomplished by the use of the snapshot application data group that extends the traditional sequential data set processing concept of generation data groups.

42 Claims, 20 Drawing figures

Previous Doc Next Doc Go to Doc#

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

☐ Generate Collection Print

L20: Entry 24 of 29 File: USPT Mar 30, 1999

US-PAT-NO: 5890165
DOCUMENT-IDENTIFIER: US 5890165 A

TITLE: Method and apparatus for automatic discovery of databases

DATE-ISSUED: March 30, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Boudrie; Robert A.	Natick	MA		
Mutalik; Madhav G.	Northboro	MA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
EMC Corporation	Hopkinton	MA			02

APPL-NO: 08/ 625014 [PALM]
DATE FILED: March 29, 1996

INT-CL: [06] G06 F 17/30

US-CL-ISSUED: 707/202; 395/182.18, 395/200.64, 707/204, 707/205
US-CL-CURRENT: 707/202; 707/204, 707/205, 709/234, 714/20

FIELD-OF-SEARCH: 395/67, 395/500, 395/182.18, 395/200.64, 395/202, 395/204, 395/205, 364/282.1, 364/182.04, 364/283.1, 364/222.81, 364/222.82, 364/283.4, 364/268.2, 707/1, 707/10, 707/101, 707/204, 707/200, 707/201, 707/205

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4714995	December 1987	Hatana et al.	364/200
☐	5109515	April 1992	Laggis et al.	395/725
☐	5185860	February 1993	Wu	395/200
☐	5276872	January 1994	Lomet et al.	395/600
☐	5561797	October 1996	Gilles et al.	395/600
☐	5594900	January 1997	Cohn et al.	707/204
	5602936	February 1997	Green et al.	382/104

☐				
☐	5644698	July 1997	Cannon	395/182.04
☐	5673382	September 1997	Cannon et al.	395/182.04
☐	5675725	October 1997	Malcom	395/182.04
☐	5687367	November 1997	Dockter et al.	707/102

OTHER PUBLICATIONS

Mosse et al. "Analysis of a Fault-Tolerance Multiprocessor Scheduling Algorithm", pp. 16-25, IEEE, Jan. 1994.
King et al, Grervicus of disaster recovery for transation processing systems, pp. 286-293, IEEE, Jul. 1990.

ART-UNIT: 276

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Corrielus; Jean M.

ATTY-AGENT-FIRM: Stretch; Maureen

ABSTRACT:

A method and apparatus for automatic discovery of databases that determines what databases are installed at a client site, interrogates each database to determine what files or tables are present and what storage is associated with each, groups the databases into the smallest atomic units that can be backed up while preserving database integrity, and presents this information to backup procedures. In a preferred embodiment, a discovery process is performed by a script for each client to be backed up. Each script causes the creation of a temporary file describing each database. The temporary files are normalized into a common format which is then passed to a grouper program that analyzes the temporary files and produces a data descriptor file describing the smallest atomic units that can be backed up while still preserving database integrity at that client site. This data descriptor file is presented to a work item generator for scheduling backup procedures. Backups of each atomic unit can then be scheduled as desired.

6 Claims, 16 Drawing figures

Previous Doc Next Doc Go to Doc#

L20: Entry 24 of 29 File: USPT Mar 30, 1999

DOCUMENT-IDENTIFIER: US 5890165 A
TITLE: Method and apparatus for automatic discovery of databases

Abstract Text (1):
A method and apparatus for automatic discovery of databases that determines what databases are installed at a client site, interrogates each database to determine what files or tables are present and what storage is associated with each, groups the databases into the smallest atomic units that can be backed up while preserving database integrity, and presents this information to backup procedures. In a preferred embodiment, a discovery process is performed by a script for each client to be backed up. Each script causes the creation of a temporary file describing each database. The temporary files are normalized into a common format which is then passed to a grouper program that analyzes the temporary files and produces a data descriptor file describing the smallest atomic units that can be backed up while still preserving database integrity at that client site. This data descriptor file is presented to a work item generator for scheduling backup procedures. Backups of each atomic unit can then be scheduled as desired.

Brief Summary Text (3):
The invention relates generally to the field of backup and recovery of databases and more particularly to the automatic discovery of databases on storage devices so that they can be grouped for backup purposes.

Brief Summary Text (9):
Such an effort is complicated by the fact that one of the most common types of database is a relational database and each relational data base has a number of files or tables created for it by the database application program. One Oracle.TM. relational database, for example, may have control files, data files, online redo log files, archive redo log files, initialization parameter files, and a file containing the Oracle.TM. code, among other files. Traditional backup planning requires that the system or database administrator know where each of these files for a given database is located on the storage units available to the system or server. FIG. 5 illustrates how a Sybase.TM.-like database, for example, DB-C, might allocate indexes C66, catalog tables C64, data tables C70, C72, into two logical groupings, system tables C60 and user database tables C68. In turn, these logical groupings may be physically allocated to two different storage units S6 and S7.

Brief Summary Text (11):
Space for each database at a client site may be grouped across several storage devices, as shown in FIG. 2. If two databases from different vendors are allocated to the same storage unit at one client site, and that unit is backed up, all the databases that have any files on that device must be completely restored to preserve database integrity. To illustrate, suppose database DB-C of FIG. 2 is allocated across storage units S6 and S7 and database DB-D is allocated across units S7 and S8. Databases DB-C and DB-D share a storage unit S7. Assume a backup copy is made of storage of units S6, and S7 for database DB-C, using an operating system-level utility. After the backup copy is made, database DB-D is updated by many transactions, but database DB-C remains as it was. If a failure occurs and the backup copy is restored to units S6 and S7, database DB-C is operational again, but database DB-D may be corrupted. This problem can occur even if both databases DB-C

and DB-D use the same vendor software.

Brief Summary Text (18):
These and other objects of the invention are achieved by a method and apparatus for automatic discovery of databases that determines what databases are installed at a client site, interrogates each database to determine what files or tables are present and what storage is associated with each, groups the databases into the smallest atomic units that can be backed up while preserving database integrity, and presents this information to backup procedures. In a preferred embodiment, a discovery process is performed by a script for each client to be backed up. Each script causes the creation of a temporary file describing each database. The temporary files are normalized into a common format which is then passed to a grouper program that analyzes the temporary files and produces a data descriptor file describing the smallest atomic units that can be backed up while still preserving database integrity at that client site. This data descriptor file is presented to a work item generator for scheduling backup procedures. Backups of each atomic unit can then be scheduled as desired.

Drawing Description Text (5):
FIG. 1d is a table illustrative of the results of the grouper program according to the method and apparatus of the present invention.

Drawing Description Text (8):
FIG. 3b is set of tables illustrating the grouping of databases according to the method and apparatus of the present invention into atomic units.

Drawing Description Text (10):
FIG. 5 is a schematic diagram of relational database tables and files as they might be allocated to storage units.

Drawing Description Text (11):
FIG. 6 is a table of environment variables that must be set according to the method and apparatus of the present invention in a preferred embodiment.

Drawing Description Text (15):
FIG. 9 is a table showing items of information required by various database vendors for access.

Detailed Description Text (5):
Database DB-A, for example, is allocated to space on three storage units S: S1, S2, and S3. Database DB-B has been allocated to two more storage units S: S4 and S5. Database DB-C has been allocated to storage units S6 and S7. Note that the fourth database, DB-D, in this example, has been allocated to storage units S7 and S8. In this example, databases DB-C and DB-D share storage unit S7.

Detailed Description Text (9):
Referring now to FIG. 3b, the overall logic of the grouping of databases into atomic units is shown. Data table 08 is a schematic representation of a normalized temporary file created by using scripts to identify which databases are present on client 10 and the storage units Sn, where n is the identity of a unit, to which they are allocated.

Detailed Description Text (10):
Referring now to FIG. 1b, the overall logic of the present invention is shown. In a preferred embodiment, the database server program must be running on client 10. A script is run to create a temporary file. In a preferred embodiment, a grouper program can then be executed to analyze table 08 and produce database descriptor file 06, which lists the atomic units contained in table 08. As shown in more detail in FIG. 1b, at step 100, a particular type of database is selected by looking for its environment variable in operating system directories. In a

preferred embodiment, the Unix operating system is used, but as will be apparent to those skilled in the art, any of a number of operating systems which permit the user of scripts or stored command sequences can be used.

Detailed Description Text (13):
Once a temporary file has been created for one data base, steps 104 through 108 are executed for each database vendor software type identified in the system environment variables. If 6 databases exist, 3 created by Oracle.TM., 1 by Sybase.TM. and 2 by Informix.TM., then 3 temporary files will be created, one for each vendor. Next, at step 109, these three files, which, in a preferred embodiment, are in ASCII format produced by the various database vendors, are normalized by a Unix or shell script that takes the three ASCII files in, and creates one output ASCII temporary file that is in uniform format. That file is generally in the format illustrated in data table 08 in FIG. 3b. That file becomes the input to a grouper program according to the method and apparatus of the present invention at step 110.

Detailed Description Text (14):
Turning now to FIG. 1c, a flow diagram of a grouper program according to the method and apparatus of the present invention is presented. At entry to grouper program 120, table 08 is provided as input. At step 122, grouper program 120, reads records from table 08 into an in-memory table with columns named

Detailed Description Text (15):
"Database.sub.-- name and "Storage.sub.-- device" as shown in FIG. 1d. At step 124, a table entry named group is established to link "grouped" elements. Initially, all the elements are ungrouped. Next, at step 126, a first loop is performed, iterating over all entries in the table and placing any which have the same database.sub.-- name into the same group. Still in FIG. 1c, next, at step 128, a second loop iterates over all entries in the table looking for any pairs which use the same storage device. When any pair uses the same storage device, the groups containing each element of the pair must be placed into the same group.

Detailed Description Text (22):
As mentioned earlier, the complexity of individual relational databases is illustrated in FIG. 5. The two different relational databases DB-C and DB-D shown earlier in FIG. 3 are schematically depicted in FIG. 5 in more detail, both as they might be structured logically and as they might be physically allocated to storage. Database DB-C, for example, has a system table C60, that contains an index C66, and several catalog tables C64a-c. This system table C60 is shown as physically allocated to storage unit S7. Addresses and other connectors C80 logically connect the system table C60 to user data C68. Each collection of user data C68, in turn may have a number of data tables C70 and C72, and C76 as well as indexes C66. In this example, user data C68 is shown allocated to storage unit S6 for database DB-C.

Detailed Description Text (23):
In the same FIG. 5, a second database, DB-D is shown, also having system tables D60, and user data D68. In this database DB-D, the location and the number of catalog tables D64, as well as data tables D72, D70 and D76 are different, as would be their contents. In this example, user data D68 for database DB-D is physically allocated to storage unit S7, while system tables D60 are allocated to storage unit S8.

Detailed Description Text (24):
The database management programs that create and manage such tables and structures as those shown in FIG. 5 vary in the way they organize the tables and files both logically and physically. The present invention backs up all the storage units associated with an atomic unit, going directly to the device or "raw" file, as it is sometimes called, itself, thus bypassing the database access in order to gain

speed.

Detailed Description Text (25):
Different vendors also have different access requirements for databases created by their respective programs. For example, as shown in FIG. 9, databases created by the Oracle.TM. database management program require a Userid and password that is designated as having the right to access such things as the Oracle.TM. v$logfile, v$datafile and v$controlfile views of the databases. (Most relational databases permit users to create "views" of the data. Views may sometimes be called virtual tables or derived tables.)

Current US Cross Reference Classification (1):
707/204

Issued US Cross Reference Classification (3):
707/204

Field of Search Class/SubClass (18):
707/204

US Reference US Original Classification (6):
707/204

US Reference Group (6):
5594900 19970100 Cohn et al. 707/204

Previous Doc Next Doc Go to Doc#

L20: Entry 25 of 29 File: USPT Mar 2, 1999

US-PAT-NO: 5878428
DOCUMENT-IDENTIFIER: US 5878428 A

TITLE: System, method, and article of manufacture for adding transactional recovery to a binary class in an object oriented system

DATE-ISSUED: March 2, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Copeland; George Prentice	Austin	TX		
Holdsworth; Simon Antony James	Andover			GB2
Smith; Stanley Alan	Austin	TX		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 559856 [PALM]
DATE FILED: November 20, 1995

INT-CL: [06] G06 F 17/30

US-CL-ISSUED: 707/103; 707/202, 707/204
US-CL-CURRENT: 707/103R; 707/202, 707/204

FIELD-OF-SEARCH: 395/614, 395/609, 395/182.18, 395/700, 395/182.13, 395/601, 395/13, 707/204, 707/103, 707/202, 364/200

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4525780	June 1985	Bratt et al.	364/200
☐	4814971	March 1989	Thatte	364/200
☐	4853842	August 1989	Thatte et al.	364/200
☐	4912708	March 1990	Wendt	371/16.3
☐	4945474	July 1990	Elliot et al.	364/200

☐	4949251	August 1990	Griffin et al.	395/609
☐	4979105	December 1990	Daly et al.	364/200
☐	4989132	January 1991	Mellender et al.	364/200
☐	4993015	February 1991	Fite, Jr.	370/16
☐	4999829	March 1991	Fite, Jr. et al.	370/16
☐	5008786	April 1991	Thatte	364/200
☐	5016162	May 1991	Epstein et al.	364/200
☐	5016243	May 1991	Fite, Jr.	370/16
☐	5043871	August 1991	Nishigaki et al.	364/200
☐	5075845	December 1991	Lai et al.	395/425
☐	5093914	March 1992	Coplien et al.	395/700
☐	5095480	March 1992	Fenner	370/94.1
☐	5111413	May 1992	Lazansky et al.	364/578
☐	5136712	August 1992	Perazzoli, Jr. et al.	395/700
☐	5163148	November 1992	Walls	707/204
☐	5185885	February 1993	Dysart et al.	395/600
☐	5261052	November 1993	Shimamoto et al.	395/200
☐	5265221	November 1993	Miller	395/725
☐	5276872	January 1994	Lomet et al.	395/600
☐	5280610	January 1994	Travis, Jr. et al.	395/600
☐	5283830	February 1994	Hinsley et al.	380/25
☐	5287453	February 1994	Roberts	395/200
☐	5291283	March 1994	Kondo et al.	348/390
☐	5295256	March 1994	Bapat	395/500
☐	5297279	March 1994	Bannon et al.	395/600
☐	5297283	March 1994	Kelly, Jr. et al.	395/650
☐	5301286	April 1994	Rajani	395/400
☐	5301316	April 1994	Hamilton et al.	395/600
☐	5303375	April 1994	Collins et al.	395/650
☐	5321841	June 1994	East et al.	395/725
☐	5325524	June 1994	Black et al.	395/600
☐	5335323	August 1994	Kolnick	395/164
☐	5341478	August 1994	Travis, Jr. et al.	395/200
☐	5343554	August 1994	Koza et al.	395/13
☐	5363313	November 1994	Lee	364/491
☐	5369702	November 1994	Shanton	380/4
☐	5369778	November 1994	San Soucie et al.	395/800
	5375227	December 1994	Akatsu et al.	395/575

☐				
☐	5379432	January 1995	Orton et al.	395/700
☐	5404506	April 1995	Fujisawa et al.	395/600
☐	5404508	April 1995	Konrad et al.	707/204
☐	5404529	April 1995	Chernikoff et al.	395/700
☐	5408649	April 1995	Beshears et al.	395/575
☐	5412774	May 1995	Agrawal et al.	395/157
☐	5414840	May 1995	Rengarajan et al.	707/204
☐	5421012	May 1995	Khoyi et al.	395/650
☐	5421015	May 1995	Khoyi et al.	395/650
☐	5437027	July 1995	Bannon et al.	395/600
☐	5446884	August 1995	Schwendemann et al.	707/204
☐	5469562	November 1995	Saether	395/182.18
☐	5481699	January 1996	Saether	395/182.13
☐	5493680	February 1996	Danforth	395/700
☐	5504883	April 1996	Coverston et al.	707/204
☐	5515502	May 1996	Wood	395/182.13
☐	5530855	June 1996	Satoh et al.	707/204
☐	5600796	February 1997	Okamura et al.	395/200.11
☐	5708776	January 1998	Kikinis	395/185.08

ART-UNIT: 271

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Coby; Frantz

ATTY-AGENT-FIRM: Walker; Mark S.

ABSTRACT:

A system, method and article of manufacture for automatically inserting transactional recoverability object service in binary classes in an information handling system employing object oriented technology, includes the steps of recognizing a constraint indicating that an object is not recoverable and generating a recoverable version of the object. One alternative for generating a recoverable version of the object includes the step of saving a first state of the object in a stream before execution of any method which might change the state of the object. Another alternative for the generating step includes the steps of creating a persistence synchronous class object, registering the persistence synchronous class object, and executing the persistence synchronous class object.

20 Claims, 5 Drawing figures

Previous Doc Next Doc Go to Doc#

First Hit Fwd Refs Previous Doc Next Doc Go to Doc#

End of Result Set

☐ Generate Collection Print

L20: Entry 29 of 29 File: USPT Oct 3, 1995

US-PAT-NO: 5455946
DOCUMENT-IDENTIFIER: US 5455946 A

TITLE: Method and means for archiving modifiable pages in a log based transaction management system

DATE-ISSUED: October 3, 1995

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Mohan; Chandrasekaran	San Jose	CA		
Narang; Inderpal S.	Saratoga	CA		

ASSIGNEE-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY	TYPE CODE
International Business Machines Corporation	Armonk	NY			02

APPL-NO: 08/ 066360 [PALM]
DATE FILED: May 21, 1993

INT-CL: [06] G06 F 11/22, G06 F 15/00

US-CL-ISSUED: 395/600; 395/488, 395/182.18, 364/DIG.1, 364/282.1, 364/285.3
US-CL-CURRENT: 707/202; 707/204, 711/161, 714/20

FIELD-OF-SEARCH: 395/600, 395/575, 364/DIG.1, 364/DIG.2

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

Search Selected Search ALL Clear

	PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
☐	4077059	February 1979	Cordi et al.	364/200
☐	4507751	March 1985	Gawlick et al.	364/DIG.2
☐	5043866	August 1991	Myre, Jr. et al.	364/DIG.1
☐	5170480	December 1992	Mohan et al.	395/600
☐	5201044	April 1993	Frey, Jr. et al.	395/575
☐	5278982	January 1994	Daniels et al.	395/600

☐	5280611	January 1984	Mohan et al.	395/600
☐	5333303	July 1994	Mohan	395/575

OTHER PUBLICATIONS

"Incremental Data Base Log Image Copy" R. A. Crus et al IBM TDB vol. 25 No. 7B Dec. 1982.

ART-UNIT: 237

PRIMARY-EXAMINER: Black; Thomas G.

ASSISTANT-EXAMINER: Homere; Jean R.

ATTY-AGENT-FIRM: Brodie; R. Bruce Saber; Paik

ABSTRACT:

A method and means for achieving files of modifiable pages in a log based phased commit transaction management system (TMS) in which those pages which have been modified since the last full or incremental backup donot require during the copy operation any modifications to the page itself but merely to a common status page. This is accomplished by management of a pair of global log sequence numbers. Comparison between a first number (ICBU.sub.-- LSN) and each data page LSN as the page is modified permits the common status page to be updated to correctly reflect the changed status. Subsequent modifications to the same page donot require amendment of the status page. The status page indicia are reset as part of the backup procedure and for ascertaining the page copy set for incremental copying. The ICBU LSN assumes one of two values as a function of the copy operation and another value for processing page modifications after the copy operation. A second number (ICRF.sub.-- LSN) is used in the restoration of a file after the file has been partially restored by a page merge in page number order from full and incremental copies. In this case, the ICRF.sub.-- LSN defines the point in the log for redo since the most recent copy was made.

6 Claims, 29 Drawing figures

Previous Doc Next Doc Go to Doc#

L20: Entry 29 of 29 File: USPT Oct 3, 1995

DOCUMENT-IDENTIFIER: US 5455946 A
TITLE: Method and means for archiving modifiable pages in a log based transaction management system

Brief Summary Text (16):
Crus et al, "Incremental Database Log Image Copy", Volume 25 IBM Technical Disclosure Bulletin, pages 3730-3732, published Dec. 1982, discloses a method for incremental copying of modified pages in a database by scanning a common status construct denominated as a space map page (SMP) and copying only those data pages having a status bit indicative (termed an incremental status change bit or ICB) of the page(s) having been changed since the last copy operation. More particularly, when a data page is first modified since a copy operation, an ICB is set both in the data and space map pages. If a data page has been modified and its ICB has already been set (indicative of previous modification since the last copy operation), then nothing is done to modify the SMP ICB. The reason an ICB bit is placed on the page is that it reduces the overhead of visiting the SMP on every update of a data page. For copying purposes, the SMP is scanned, each page whose ICB has been set is copied, and the ICB counterparts on both the data page and the SMP are then reset.

Brief Summary Text (22):
The invention utilizes constructs available as part of a log based phased commit TMS. These include a space map page (SMP) and page update status bit (ICB), a header page, a database control block (DBCB), the log, two types of LSN's (ICBU.sub.-- LSN; ICRF.sub.-- LSN), and a system catalog. The SMP includes an ICB for each data page, the header page stores the ICBU.sub.-- LSN for the database, the DBCB maintains the ICBU.sub.-- LSN in virtual storage, the system catalog stores the ICRF.sub.-- LSN for the database.

Brief Summary Text (26):
The ICRF.sub.-- LSN denotes the LSN from which the log would have to be scanned to identify log records which must be REDOne (REDO ops) to recover the database after reloading the latest full copy and subsequent incremental copies.

Detailed Description Text (10):
Manger 22 regulates the data structures and storage space of database 13. It provides record level access to transactions and page level access to system utilities such as loading, copying, or recovering a database. The buffer manager moves pages between the database 13 and the buffer pool 18. The latch manager provides short term serialization (shared/exclusive) to pages being read or modified in the buffer pool 18. The log manager 24 generates the log records, assembling them in numbered sequences in log buffer 20, and writing them to log 21. The recovery manager 25 utilizes the log records to return the database to support transaction level recovery while the concurrency manager 26 implements locks via lock table 30.

Detailed Description Text (16):
First, the data structures in the data base and in virtual storage, the logging

protocol, and the latching protocol are described. Second, the processing relating to the copy operation, a transaction's update to set the ICB (if required), and the recovery of a data base using the backup copy is set out. Third, the processing involved in the rollback of an copy, in case its execution is interrupted by a failure is explained. Fourth, the manner of usage of the method of the invention is discussed where it is possible to copy the data directly from DASD storage instead of going through the buffer pool associated with a DB2 like system. Lastly, extensions of the method of the inventions to a multisystem transaction processing system with shared DASD storage are described.

Detailed Description Text (18):
For purposes of this invention, a data base includes data pages and ancillary constructs. In this regard, a file of pages is deemed to be a functional equivalent to a data base. The ancillary constructs include pages in the data base for tracking the allocation status etc. A page containing user data is called a "data page", a system-owned page tracking the allocation and space availability status of data pages is called a "space map page" (SMP), and a system-owned page containing system related information is called a "header page".

Detailed Description Text (19):
As in DB2, a large table can be divided into many partitions each of which is a separate operating system file. Each partition would have a header page, one or more space-map pages and numerous data pages. Archive copies include copies of not only data pages but also those of SMPs and the header page.

Detailed Description Text (23):
Associated with each archive copy is an copy roll forward LSN (ICRF.sub.-- LSN). During media recovery, this is the LSN from which the log would have to be scanned to identify log records whose updates might have to be REDOne to recover the data base after reloading the relevant archive copies (the latest full copy and any subsequent incremental copies). ICRF.sub.-- LSN is remembered in a system catalog along with the information such as the name of the file which contains the archive copy, and whether a full or incremental copy was taken.

Detailed Description Text (41):
This section is directed to (a) the copy operation, (b) update of a data page by a transaction and, (c) the recovery of the data base after a media failure. The rollback processing in case copy operation's execution is interrupted by a failure is also described.

Detailed Description Text (100):
Referring now to FIG. 3, there is shown a logical file and page organization as used in the invention. More particularly, a file or data base includes a header page, a space map page (SMP), and a plurality of data pages (d-pages 1-4). The data, arbitrarily named "file A", is loaded into the database 13 in the DASD external storage under control of a CPU controlled load operation. To save log space and logging overhead, such loading is performed with the logging suspended. Also, no concurrent updates are permitted during execution of the loading operation. As part of the operation, page numbers are sequentially assigned to each page. The header page is number page 0, the SMP is page 1, while d-page 1-4 are numbered as pages 2-5. Now, the page numbers merely define a local page sequence and should NOT be confused with the page.sub.-- LSN's. The latter are pointers embedded in each page when written through to DASD storage after being logged which denote the position of the REDO/UNDO records in the log containing the most recent page modification. Initially, the page.sub.-- LSN's are set to 0. Likewise, the ICB's in the SMP for all the d-pages are set to 0 as is the ICBU.sub.-- LSN in the header page. Lastly, since no backup exists for file A, there is no entry in the system catalog.

Detailed Description Text (101):

As seen in FIG. 3, the system catalog broadly contains a location pointer and other archive or backup copy information for an either a full or incremental copy of file A and other files. Typically, such archive copies would be stored on an auxiliary store such as an automated tape library. Any recovery use of the backup copy would require access of the tape library and staging therefrom to DASD storage.

Detailed Description Text (117):
Significantly, if the ICBU.sub.-- LSN value had remained at 100, then the update to d-page 1 would NOT have resulted in ICB1 being set to "1". Thus, subsequent IICs would not copy d-page 1 with the value "CI". Any completion of a subsequent IC would result in an ICRF.sub.-- LSN>504. In the case of any loss of file A after the subsequent IC, the restoration of backup copies using an ICRF.sub.-- LSN >504 would cause the update of LSN 504 to be missed. This explains the need for setting the ICBU.sub.-- LSN to a maximum value DURING THE IC while a new non-maximum value is being established. Setting ICBU.sub.-- LSN to a non-max value AFTER completing the IC including latching, copying, and unlatching all pages in the copy set is required, otherwise every update to any page in the file would cause access to the SMP for setting the corresponding ICB or checking whether the ICB is already set. The latter is considered wasteful.

Current US Cross Reference Classification (1):
707/204

CLAIMS:

5. A method for archiving a database of modifiable pages in a phased commit transaction management system (TMS) having a log, a processor, and a storage subsystem in which pages stored therein are staged to and from the processor,

the processor being responsive to each transaction and executing selective modifications to predetermined ones of the pages, each modification to a page state being recorded in the log and assigned an ascending log sequence number (page.sub.-- LSN), said page.sub.-- LSN being recorded on the page,

comprising the steps of:

(a) initializing a copy operation for a partition of the database by

(1) invoking a database control block (DBCB) for said partition, said partition having a header page, at least one data page, and at least one space map page (SMP) including a status bit (ICB) for each data page indicative as to whether the data page has been modified since execution of a prior copy operation,

(2) atomically setting an LSN of a first kind (ICBU.sub.-- LSN) to a maximum value in the DBCB, setting an LSN of a second kind (ICRF.sub.-- LSN) to a value set by the current end.sub.-- of.sub.-- log LSN, and recording the ICRF.sub.-- LSN in a system catalog, and

(3) ascertaining the set of data page's subject to copying for each SMP by resetting all ICB's of data pages indicative of being updated since the last copy operation and recording all of the ICB's so reset in the log; and

(b) executing said copy operation by

(1) for each SMP, latching, copying, and unlatching each data page identified in step (a)(3), and

(2) updating the ICBU.sub.-- LSN in the DBCB and the header page to the current end.sub.-- of.sub.-- log LSN value,

(3) committing the operation, and

(c) responsive to each page modification, setting the counterpart ICB in the status page if the page.sub.-- LSN<ICBU.sub.-- LSN and record the action in the log.

Previous Doc Next Doc Go to Doc#